LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-418522

Contention-free Routing for Shift-based Communication in MPI Applications on Large-scale Infiniband Clusters

Adam Moody

October 23, 2009

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Contention-free Routing for Shift-based Communication in MPI Applications on Large-scale Infiniband Clusters

Adam Moody
Livermore Computing,
Lawrence Livermore National Laboratory,
Livermore, CA 94551, USA
moody20@llnl.gov

September 9, 2008

1 Shift-based communication in MPI

Shift-based communication can be defined as follows. For a set of N nodes assigned to a job, assign each node an ID from 0 to $N-1$. Shift-based communication involves $N-1$ steps. Let D denote the current step, and let D iterate from 1 to $N-1$. Then in step D all nodes choose nodes to send to and receive from such that a node with $ID = I$ sends to the node with $ID = (I + D)\%N$ and receives from the node with $ID = (I - D + N)\%N$, where % denotes modulo division. Figure 1 illustrates the communication patterns for various steps in shift-based communication.

Shift-based communication patterns enable all nodes in a job to send and receive data simultaneously in all steps. Many MPI operations employ shift-based communication patterns for this reason [1, 2]. This includes large message collective algorithms, such as those typically used to implement large message Allgather and Alltoall collectives. It also includes small message collective algorithms, such as pair-wise exchange, barrier dissemination [3], and Bruck's index algorithm [4]. Although the small message algorithms typically pack messages such that each node does not send to or receive from every other node directly, the communication patterns they do execute correspond to particular steps in shift-based communication. Common point-to-point message patterns also can benefit from efficient shift-based routing, such as nearest-neighbor exchanges in domain decomposition codes. Supporting efficient shift-based communication within a job thus provides good performance for a number of common MPI operations.

2 mpiGraph: Visualizing contention in shift-based communication

The default routing employed in large-scale Infiniband [5] clusters leads to heavy contention for shift-based communication patterns. A benchmark called *mpiGraph* [6] can be used to illustrate this problem. The MPI tasks in this benchmark execute a shift-based communication pattern. In each step of the shift, each task measures its receive bandwidth. After executing all steps, each task has a measurement of its bandwidth from every other task. This data is then arranged in a 2-D matrix such that the data at row a and column b represents the receive bandwidth task a measured when it received from task b. The exact diagonal in this matrix represents a task receiving from itself, which is not measured. The minimum, average, and maximum bandwidths of all values in the matrix are computed. The 2-D matrix is then converted into a gray-scale bitmap image by representing each entry as a pixel. The maximum bandwidth in the matrix is set to a pixel value of 255 (pure white), while all other values are scaled to a pixel value of 0 (pure black) depending on their percentage of the maximum bandwidth.

Figure 2 shows the mpiGraph images generated for four large-scale Infiniband clusters: Hype, Zeus, Rhea, and Atlas. Next to each image is a histogram of the pixel values. The horizontal axis in the histogram plots the pixel value, which ranges from 0 (black) on the left to 255 (white) on the right. The vertical axis plots the count of the number of pixels of a given value in the image.

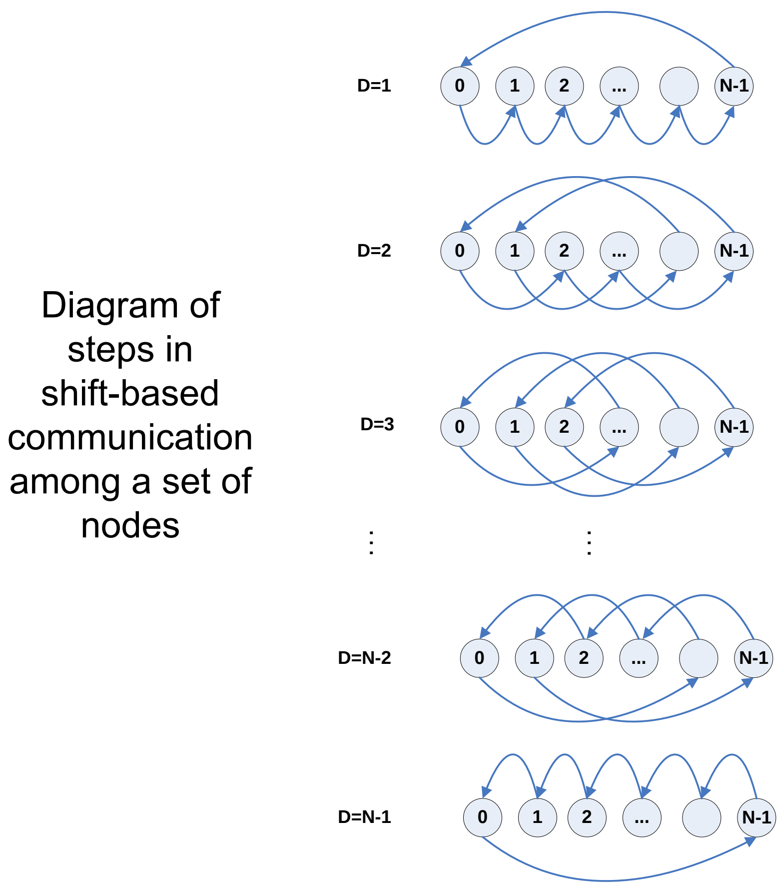

Figure 1: Shift-based communication

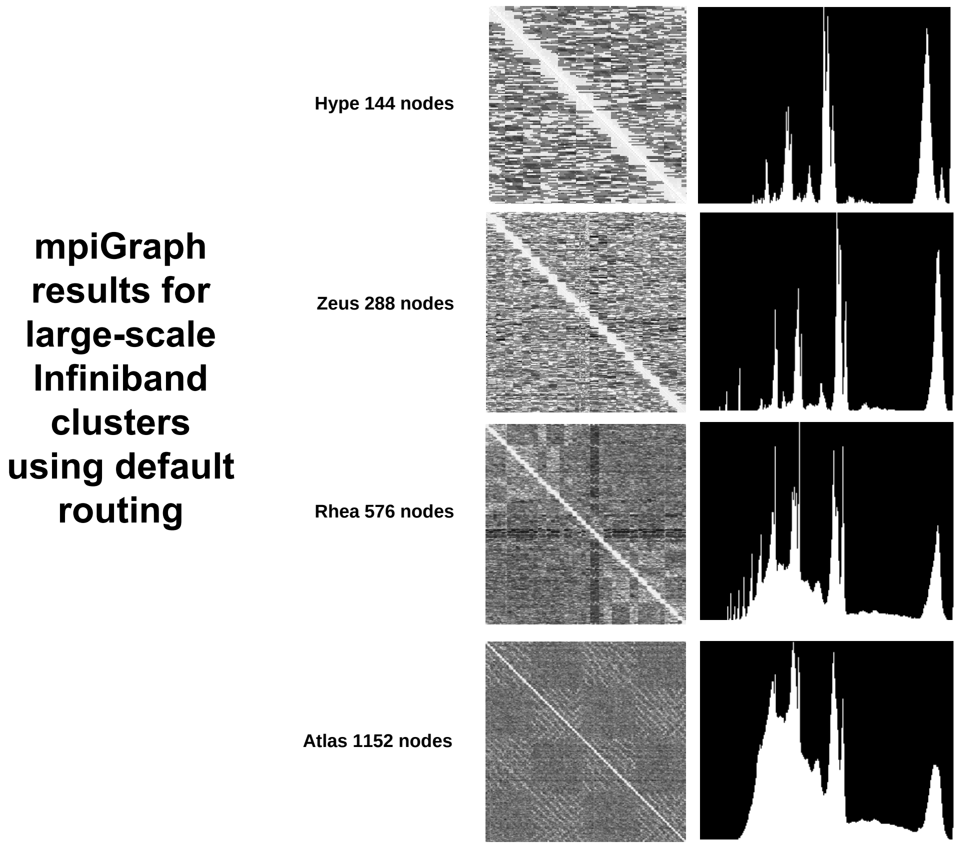

Figure 2: mpiGraph images for four large-scale clusters

While the mpiGraph images look to be noisy at first glance, the same image can be reproduced reliably run after run. The histogram provides a clue as to why this is the case. Looking at the histogram for Hype, note the distinct spikes around certain pixel values. This implies the interconnect has a tendency to favor distinct bandwidth values. As we'll soon show, these distinct bandwidth values arise from the contention in the network.

The minimum, maximum, and average bandwidths recorded during the mpiGraph tests for each cluster are shown in Table 1. These bandwidths were measured by sending 16 KB messages on Infiniband 4x SDR links using MVAPICH-0.9.7. There is a wide spread between the minimum and maximum values, with the minimum bandwidth typically coming in at one tenth of the maximum bandwidth. Also note how the average bandwidth is considerably lower than the maximum, and note how the average steadily declines as the cluster size increases. All of these trends negatively impact performance and scalability on large-scale Infiniband clusters. In fact, on Atlas, the average bandwidth drops to less than one-half of the maximum, such that the network delivers less than half of its peak performance on average.

To determine what effect routing contention has on the mpiGraph results, a script called *routeGraph* was written to estimate the mpiGraph images based on the contention found in the Infiniband routing tables. To do this, routeGraph is fed the Infiniband LID forwarding tables (output from the `/usr/sbin/dump_lfts.sh` command), the network connectivity tables (output from the `/usr/sbin/ibnetdiscover` command), and the set of nodes mpiGraph ran on as input. The script then iterates through the steps of the shift-based communication that mpiGraph executes. For a given step, routeGraph determines which node each node sends to. For each send, it

Table 1: mpiGraph statistics for four large-scale clusters

Cluster	Nodes	min MB/s	avg MB/s	max MB/s
Hype	144	159	503	796
Zeus	288	57	435	713
Rhea	544	72	347	741
Atlas	1152	96	340	736

traces the route from sender to receiver through the network and increments each link used in that route by one. After tracing all routes for all sends in a given step, each route is traced again to identify the link on that route with the highest count. This link represents the bottleneck in this route, and the count is the number of routes using that link simultaneously. The bandwidth for the send along that route is thus set to be the peak link bandwidth divided by the number of routes using the bottleneck link simultaneously. This defines the bandwidth for the receiver of the send, and hence, the estimated image pixel value for the corresponding receiver/sender node pair. The bandwidths and corresponding pixel values are computed for all node pairings in a step in this way, and then the link counts are cleared and the next step in the shift is processed. After processing all steps, routeGraph outputs the estimated mpiGraph image. Like mpiGraph, a histogram of pixel values is also computed for the routeGraph image.

The routeGraph image and histogram created from Hype's routing tables is shown in Figure 3. Note that the details of the image are reproduced quite precisely, and the spikes in the mpiGraph histogram also show up in the routeGraph histogram. (The rightmost spike in the mpiGraph histogram does exist in the routeGraph histogram, but it is difficult to distinguish because it is located at the far right edge of the image.) This result verifies that the spikes in the histogram and the generally "noisy" texture in Hype's mpiGraph image are due to network contention.

3 Contention-free routing for two switch level clusters

3.1 Cluster wiring

Large-scale Infiniband networks are often constructed from a number of smaller $2X$-port bidirectional, crossbar switches, where X is some positive integer greater than 1. A crossbar switch has two important properties. First, data coming in any port can be switched to go out any other port. Second, the ports can be paired up in all possible combinations without contention. The switch ports are numbered 0 to $2X-1$, as shown in the top portion of Figure 4. Typically these switches are wired together in layers to build bigger networks. With this layering, it helps to distinguish between ports connected to lower layers, *down-ports*, and ports connected to higher layers, *up-ports*. As such, we designate half of the ports, 0 to $X-1$, as down-ports with indices 0 to $X-1$, and the other half, X to $2X-1$, as up-ports with indices 0 to $X-1$, as shown in the bottom portion of Figure 4. The networks on LLNL clusters are constructed using 24-port switches, such that $X = 12$.

Wiring Rule 1. Consider a cluster with two switch levels constructed from $2X$-port switches, as in Figure 5. This network is wired such that X nodes are attached to each leaf-level switch, each of which has X up-ports linked to the down-ports of X 2nd-level switches. If the leaf-level switches are labeled $S1.0$ through $S1.X-1$, and the 2nd-level switches are labeled $S2.0$ through $S2.X-1$, then a simple rule describes this wiring. Up-port b on switch $S1.a$ should be wired to down-port a on switch $S2.b$, i.e., $\forall a, b \in [0, X-1]$,

$$S1.a : b \leftrightarrow S2.b : a \qquad (1)$$

3.2 Node selection

Given a cluster with two switch levels wired as described in Wiring Rule 1, different constraints can be placed on the resource manager to allocate nodes to a job such that contention-free routing can be implemented for shift-based communication. Here we choose the following constraint:

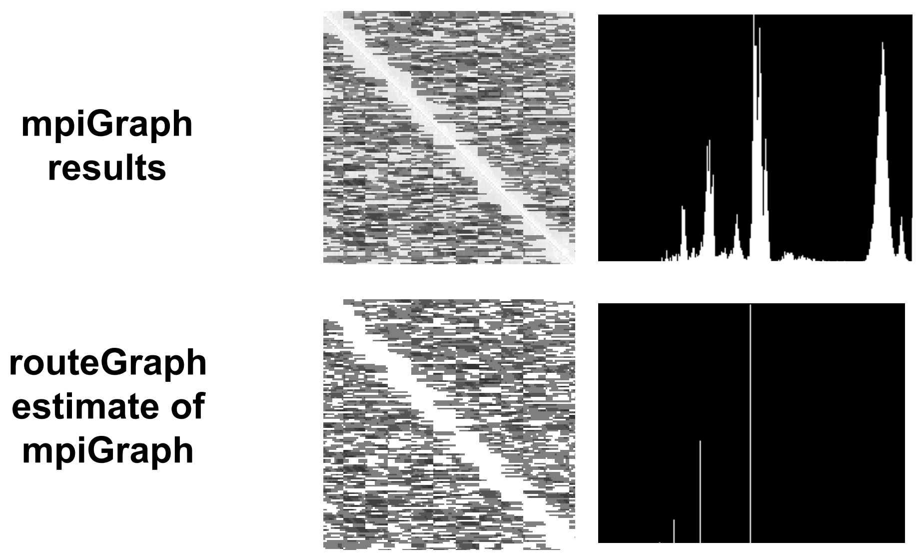

Figure 3: mpiGraph results and routeGraph estimate for Hype

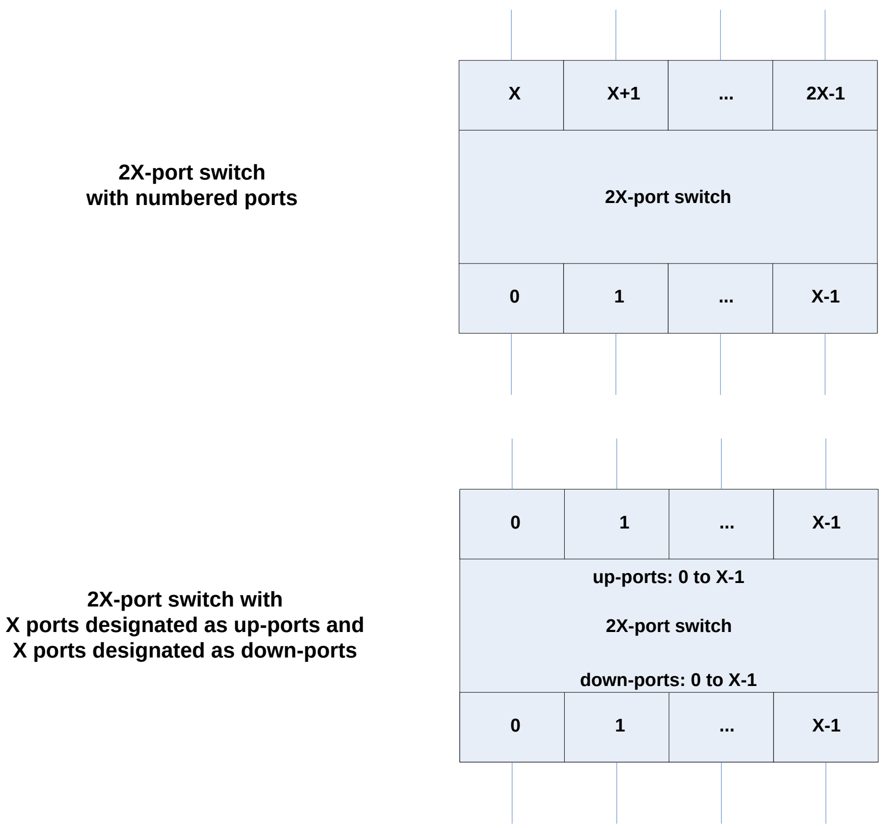

Figure 4: Building block: $2X$-port bidirectional, crossbar switch

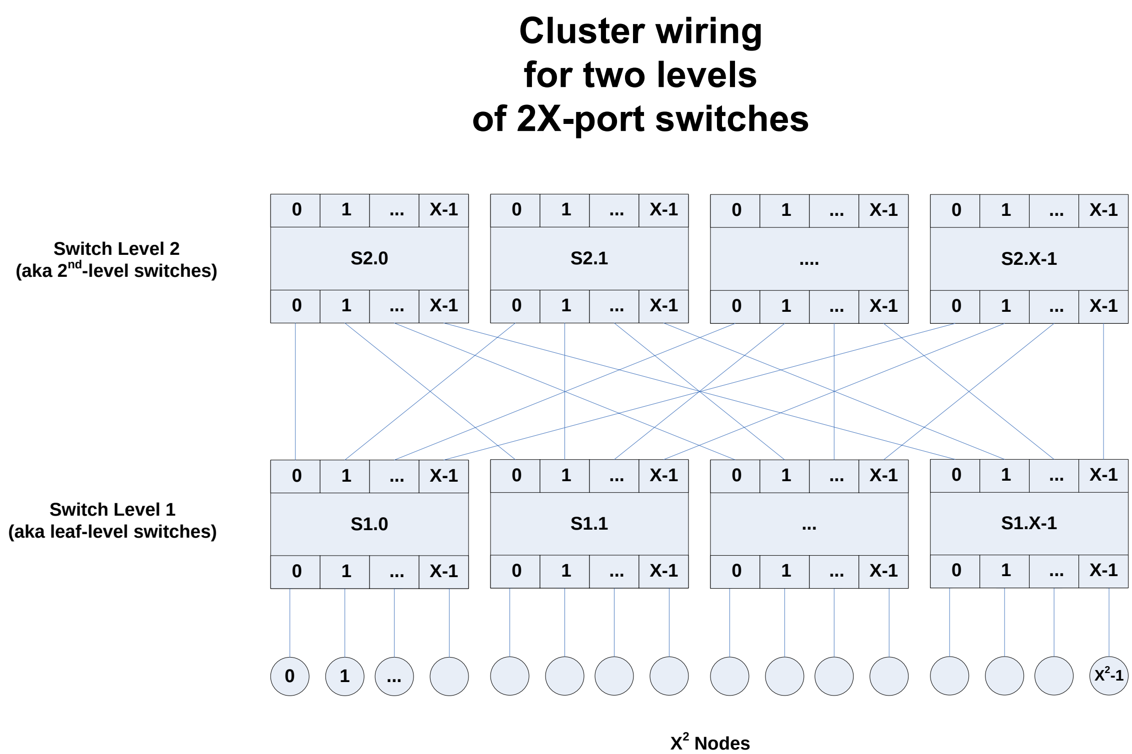

Figure 5: Cluster wiring for two switch levels of $2X$-port switches

Node Selection Constraint 1. Pick nodes such that the maximum number of nodes, Y, on any leaf switch, is less than or equal to some number, M in the range $[1, X]$, chosen such that M evenly divides the number of nodes in the job, N.

In practice, one can determine a value for M as follows. For a job of N nodes, find all integers in the range $[1, X]$ which divide N evenly, and then set M to be the largest of such integers. The resource manager should then assign no more than M nodes from any one leaf switch to the job.

3.3 Routing

Infiniband switches are destination routed, meaning a switch makes a decision to route a packet out a particular port based solely on the destination address of the packet. Further, this decision is determined via a routing table programmed into the switch, so that the switch does a simple table lookup to get the index of the port to use to forward packets to a given destination address. With this in mind, we apply the following routing algorithm.

Routing Algorithm 1. First, assign node IDs to the nodes in the job from 0 to $N-1$ ordered left to right by leaf-level switch and then port number. Following the order of the node IDs, cyclically assign indices to the nodes in the job from 0 to $M-1$. Call this the *M-index* of the node. Because M divides N evenly, there will be N/M full cycles of M-indices. Note that the M-index can be directly computed from the node ID, I:

$$Mindex(I) = I\%M \tag{2}$$

Figure 6 shows an example of labeling node IDs and M-indices for a 9-node job on 8-port switches. Here, the maximum number in the range $[1, X = 4]$ which divides 9 evenly is 3, so we set $M = 3$ and assign no more than 3 nodes from each leaf-level switch to the job such that $1 \leq (Y = 3) \leq (M = 3) \leq (X = 4)$ and $(N = 9)\%(M = 3) = 0$.

After the node IDs have been assigned, program each leaf-level switch to route packets out the corresponding down-port if the destination node is attached to a down-port. Otherwise, route packets up to a 2nd-level switch using the up-port that has the same index as the M-index of the destination node. That is, if D is the node ID of the destination node, select the up-port index, u, by the following rule,

$$S1 \rightarrow S2 : u = Mindex(D) = D\%M \tag{3}$$

Note that the up-port index selected will be in the range $[0, M-1]$, and $M \leq X$, so that M is less than or equal to the number of up-ports on the switch.

Finally, program each 2nd-level switch to route packets via the down-port that connects directly to the leaf-level switch that contains the destination node.

This is a constrained version of minimum-hop routing in which we are picking a particular up-path among a set of many possible up-paths.

Routing Algorithm 1 applied to a cluster wired according to Wiring Rule 1 ensures that:

Property 1. $\forall a \in [0, X-1]$, a 2nd-level switch with label $S2.a$ receives only packets destined for a node with M-index equal to a.

Proof of Property 1. According to Wiring Rule 1, down-port index b on a 2nd-level switch with label $S2.a$ is connected to up-port index a on the leaf-level switch with label $S1.b$ for all $a, b \in [0, X-1]$. Thus, for a fixed a, on the 2nd-level switch with label $S2.a$, each down-port with index b for all $b \in [0, X-1]$ is connected to an up-port with index a on some leaf-level switch, namely the switch with label $S1.b$. Thus, the 2nd-level switch with label $S2.a$ is only connected to leaf-level switches at up-port index a.

Furthermore, according to the $S1 \rightarrow S2$ up-port selection rule in Routing Algorithm 1, to route packets up to 2nd-level switches, leaf-level switches select the up-port index that matches the M-index of the destination node. Thus, only packets destined for a node with M-index equal to a will select up-port index a. Since the 2nd-level switch with label $S2.a$ is only connected to leaf-level switches at up-port index a, it only receives packets destined for a node with M-index equal to a.

Finally, since these wiring and routing rules hold for all $a \in [0, X-1]$, we conclude that: $\forall a \in [0, X-1]$, a 2nd-level switch with label $S2.a$ receives only packets destined for a node with M-index equal to a. $\square$

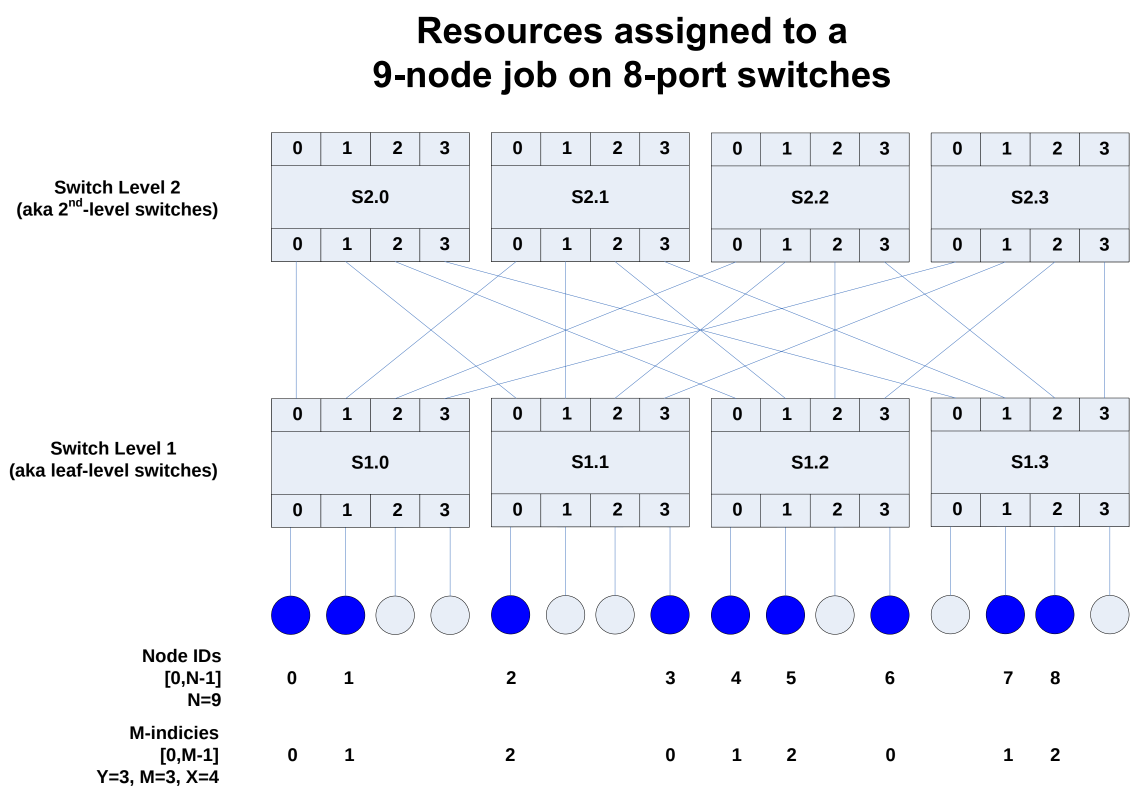

Figure 6: Labeling nodes assigned to a 9-node job on 8-port switches

3.4 Proof of no contention

We now prove that Node Selection Constraint 1 and Routing Algorithm 1 enable contention-free routing for shift-based communication within a job on a cluster wired according to Wiring Rule 1.

There is no contention from a node up to its leaf-level switch, since it is the only node which can send on the link that connects that node to its leaf-level switch. There is no contention from the leaf-level switch down to a node, since the shift-based algorithm ensures that exactly one node will ever send to a given node in a given step. Thus, there is no contention between the nodes and the leaf-level switches.

For there to be contention going up from a leaf-level switch to a 2nd-level switch, two nodes attached to the same leaf must send to destination nodes which are routed out the same up-port. As the leaf-level switches select the up-port index that matches the M-index of the destination, this means two nodes on the same leaf must send to nodes with the same M-index. However, the nodes on a given leaf-level switch have a consecutive set of node IDs. At a given step in the shift-based communication, each node sends to a node D hops up relative to its node ID. Since the nodes on a leaf have consecutive node IDs and because they all send D hops up, the corresponding set of destination nodes they collectively send to must also have consecutive node IDs. A set of consecutive node IDs is assigned consecutive M-indices in the cyclic range $[0, M-1]$. Since there are no more than Y nodes on a leaf, and $Y \leq M$, there can be no more than M sender nodes on a given leaf. Hence, there can be no more than M corresponding destination nodes. As such, there are not enough destination nodes to wrap the M-index range of $[0, M-1]$, which means that each destination node must have a unique M-index in each step of the shift. Since it is this M-index that is used to select the up-port, packets from the nodes sending on the same leaf must select unique up-ports.

For there to be contention going down from a 2nd-level switch to a leaf-level switch, two packets coming in to a given 2nd-level switch must be routed down to the same leaf-level switch. By Property 1, all packets delivered to a given 2nd-level switch from leaf-level switches are headed to destination nodes with the same M-index. Thus, for there to be contention, two nodes sending to destination nodes with the same M-index must be sending to nodes on the same leaf-level switch. Since shift-based communication ensures that only one node will send to each destination node in a given step, for contention to occur, there must be two destination nodes on a leaf-level switch with the same M-index. However, this can not be since M-indices are assigned consecutively in the cyclic range $[0, M-1]$, and there are no more than $Y \leq M$ nodes attached to any leaf. Thus, there are not sufficient destination nodes on any leaf to fully wrap the M-index range.

As there is no contention between the nodes and leaf-level switches, and there is no contention between the leaf-level switches and the 2nd-level switches, Node Selection Constraint 1 and Routing Algorithm 1 implement contention-free routing for shift-based communication within a job on a cluster wired according to Wiring Rule 1.

3.5 Extending to a full two switch level cluster

In a typical production cluster, the goal is to maximize the number of nodes for a given number of switches. For such clusters, the up-ports on the 2nd-level switches would not be left idle. Instead, each of these ports would be assigned as another down-port connected to an additional leaf-level switch filled with an additional X nodes. Although, not proved here, the above argument can be naturally extended to such clusters, i.e., two switch level clusters containing $2X$ leaf-level switches and X 2nd-level switches.

In this case, the 2nd-level switches have no up-ports, but instead all $2X$ ports are designated as down-ports with with indices 0 to $2X-1$. Then, we just extend Wiring Rule 1 to say: $\forall a \in [0, 2X-1]$ and $\forall b \in [0, X-1]$, connect $S1.a : b \leftrightarrow S2.b : a$. With this extension, it can be shown using a similar argument as above that Node Selection Constraint 1 and Routing Algorithm 1 implement contention-free routing for shift-based communication within a job on such a cluster.

Taking this further, some clusters may have fewer than X 2nd-level switches. Such clusters connect each leaf-level switch to the same 2nd-level switch multiple times, e.g., X leaf-level switches and $X/2$ 2nd-level switches where each leaf-level switch connects twice to each 2nd-level switch. In this case, each 2nd-level switch has multiple down-ports available to reach a given leaf-level switch. In order to pick a down-port, we assign each down-port the same M-index that is assigned to the up-port on the leaf-level switch that connects to it. Then, when routing packets down, we select the down-port that matches the M-index of the destination node. Although not proved here, with this additional routing condition, the above argument can be extended to show that this implements contention-free routing for shift-based communication within a job on such a cluster.

4 Contention-free routing for three switch level clusters

4.1 Cluster wiring

Wiring Rule 2. We can extend the pattern described by Wiring Rule 1 to wire a cluster with three switch levels. We'll use the two switch level cluster shown in Figure 5 as a building block. We'll refer to each two switch level building block and the nodes it contains as a "2nd-level leaf".

Now to build a three switch level cluster, wire X such blocks to a set of X^2 3rd-level switches as shown in Figure 7. In this configuration, there are X^2 switches at each switch level. Label each switch at a given level 0 through X^2-1 from left to right. Call this the switch ID. Then, translate the switch ID into a dotted label notation S$L.a.b$, where L is the level number, $a = \lfloor ID/X \rfloor$, and $b = ID\%X$. With this labeling, the wiring can be described by two simple rules. Up-port c on switch $S1.a.b$ should be wired to down-port b on switch $S2.a.c$, and up-port b on switch $S2.a.c$ should be wired to down-port a on switch $S3.b.c$, i.e., $\forall a, b, c \in [0, X-1]$,

$$S1.a.b : c \leftrightarrow S2.a.c : b \tag{4}$$

$$S2.a.c : b \leftrightarrow S3.b.c : a \tag{5}$$

Note that with this wiring, each 2nd-level switch in a 2nd-level leaf has exactly one connection to each leaf-level switch contained in the 2nd-level leaf. Thus, starting from any 2nd-level switch in a 2nd-level leaf, a packet may reach any node contained in the 2nd-level leaf via a hop through a single leaf-level switch.

4.2 Node selection

Given a cluster with three switch levels wired as described in Wiring Rule 2, different constraints can be placed on the resource manager to allocate nodes to a job such that contention-free routing can be implemented for shift-based communication. Here we choose the following constraint:

Node Selection Constraint 2. Pick nodes such that the maximum number of nodes on any leaf-level switch, Y_1, is less than or equal to some number, M_1 in the range $[1, X]$, and the maximum number of nodes on any 2nd-level leaf, Y_2, is less than or equal to some number, $M_1 * M_2$, where M_2 is in the range $[1, X]$ and the product $M_1 * M_2$ evenly divides the number of nodes in the job, N.

In practice, one can determine values for M_1 and M_2 as follows. For a job of N nodes, find all integers in the range $[1, X]$ which divide N evenly. For each such integer, m_1, find all integers, m_2, in the range $[1, X]$ which divide N/m_1 evenly. For all qualifying pairs of integers (m_1, m_2), find the pair such that $m_1 * m_2$ is largest and set $M_1 = m_1$ and $M_2 = m_2$. Then, the resource manager should assign no more than M_1 nodes from any one leaf-level switch to the job, and it should assign no more than $M_1 * M_2$ nodes from any one 2nd-level leaf to the job. After selecting a set of nodes according to this constraint, we apply the following routing algorithm.

4.3 Routing

Routing Algorithm 2. Assign node IDs to the nodes in the job from 0 to $N-1$ ordered left to right by 2nd-level leaf, then leaf-level switch, and then port number. Then, assign M_1*-indices* and M_2*-indices* to each node computed as

$$M_1 index(I) = I\%M_1, \tag{6}$$

$$M_2 index(I) = \lfloor I/M_1 \rfloor \% M_2, \tag{7}$$

where I is the node ID. Define the tuple $(M_2 index, M_1 index)$ as the M*-index* of a node.

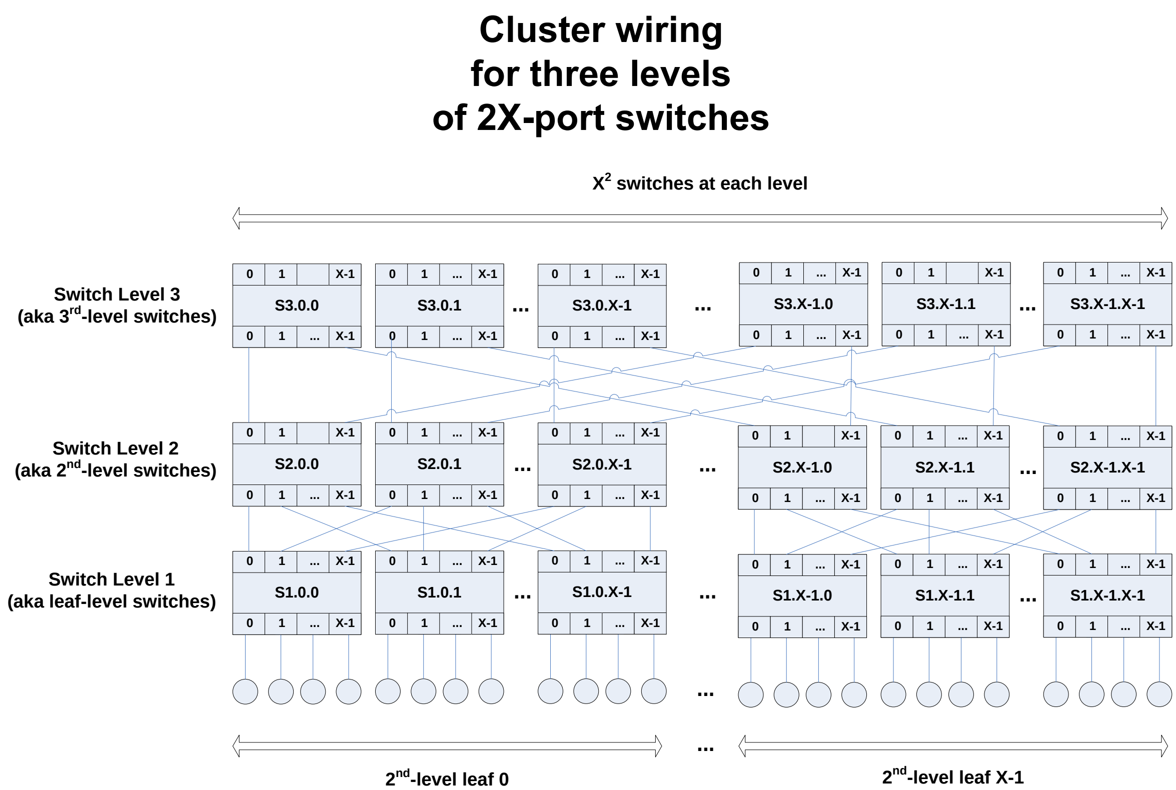

Figure 7: Cluster wiring for three switch levels of $2X$-port switches

With this definition, as one iterates through the nodes in order of the node IDs, the M_1-index increments with each node and cycles after every M_1 nodes, while the M_2-index increments each time M_1 cycles and the M_2-index cycles after every $M_1 * M_2$ nodes.

Now, similar to Routing Algorithm 1, program each leaf-level switch to route packets out the appropriate down-port if the destination node is attached to a down-port. Otherwise, route packets up to a 2nd-level switch using the up-port with an index matching the M_1-index of the destination node.

Then, program 2nd-level switches to route packets out the appropriate down-port if the destination node is attached to a leaf-level switch reachable by a down-port. Otherwise, route packets up to a 3rd-level switch using the up-port index that matches the M_2-index of the destination node.

That is, if D is the node ID of the destination node, select the up-port indices according to the following rules,

$$S1 \rightarrow S2 : u_1 = M_1 index(D) = D\%M_1, \tag{8}$$

$$S2 \rightarrow S3 : u_2 = M_2 index(D) = \lfloor D/M_1 \rfloor \%M_2. \tag{9}$$

Finally, program 3rd-level switches to route packets down via the down-port that connects to a 2nd-level switch in the 2nd-level leaf that contains the destination node.

As before, this is a constrained version of minimum-hop routing in which we are picking a particular up-path among a set of many possible up-paths.

Routing Algorithm 2 applied to a cluster wired according to Wiring Rule 2 ensures the following properties:

Property 2. $\forall a, c \in [0, X-1]$, a 2nd-level switch with label $S2.a.c$ receives only packets destined for a node with M_1-index equal to c from its leaf-level switches.

Property 3. $\forall b, c \in [0, X-1]$, a 3rd-level switch with label $S3.b.c$ receives only packets destined for a node with (M_2,M_1) M-index equal to (b,c).

Property 4. $\forall a, c \in [0, X-1]$, a 2nd-level switch with label $S2.a.c$ receives only packets destined for a node with M_1-index equal to c.

Proof of Property 2. According to Wiring Rule 2, down-port index b on a 2nd-level switch with label $S2.a.c$ is connected to up-port index c on the leaf-level switch with label $S1.a.b$ for all $a, b, c \in [0, X-1]$. Thus, for a fixed a and c, on the 2nd-level switch with label $S2.a.c$, each down-port with index b for all $b \in [0, X-1]$ is connected to an up-port with index c on some leaf-level switch, namely the switch with label $S1.a.b$. Thus, the down-ports of the 2nd-level switch with label $S2.a.c$ are only connected to leaf-level switches at up-port index c.

Furthermore, according to the $S1 \rightarrow S2$ up-port selection rule in Routing Algorithm 2, to route packets up to 2nd-level switches, leaf-level switches select the up-port index that matches the M_1-index of the destination node. Thus, only packets destined for a node with M_1-index equal to c will select up-port index c. Since the down-ports of the 2nd-level switch with label $S2.a.c$ are only connected to leaf-level switches at up-port index c, switch $S2.a.c$ only receives packets destined for a node with M_1-index equal to c from those leaf-level switches.

Finally, since these wiring and routing rules hold for all $a, c \in [0, X-1]$, we conclude that: $\forall a, c \in [0, X-1]$, a 2nd-level switch with label $S2.a.c$ receives only packets destined for a node with M_1-index equal to c from its leaf-level switches. □

Proof of Property 3. According to Wiring Rule 2, down-port index a on a 3rd-level switch with label $S3.b.c$ is connected to up-port index b on the 2nd-level switch with label $S2.a.c$ for all $a, b, c \in [0, X-1]$. Thus, for a fixed b and c, on the 3rd-level switch with label $S3.b.c$, each down-port with index a for all $a \in [0, X-1]$ is connected to an up-port with index b on some 2nd-level switch, namely the switch with label $S2.a.c$. Thus, the 3rd-level switch with label $S3.b.c$ is only connected to 2nd-level switches at up-port index b.

Furthermore, according to the $S2 \rightarrow S3$ up-port selection rule in Routing Algorithm 2, to route packets up to 3rd-level switches, 2nd-level switches select the up-port index that matches the M_2-index of the destination node. Thus, only packets destined for a node with M_2-index equal to b will select up-port index b. Since the 3rd-level switch with label $S3.b.c$ is only connected to 2nd-level switches at up-port index b, it only receives packets destined for a node with M_2-index equal to b from those switches.

In addition, for a fixed b and c, note that the 3rd-level switch with label $S3.b.c$ is only connected to 2nd-level switches with label $S2.a.c$ where $a \in [0, X-1]$. By Property 2, we know that all packets a 2nd-level switch with label $S2.a.c$ receives from its leaf-level switches are destined for a node with an M_1-index equal to c. Additionally, the only packets a 2nd-level switch sends up to 3rd-level switches are those it receives from its leaf-level switches. (According to Routing Algorithm 2, if a 2nd-level switch receives a packet from a 3rd-level switch, it is because that 2nd-level switch is part of the 2nd-level leaf that contains the destination node, in which case, the 2nd-level switch will forward the packet down to the appropriate leaf-level switch that contains the node. No packet received by a 2nd-level switch from a 3rd-level switch will be forwarded back up to a 3rd-level switch.) Thus, the only packets a 3rd-level switch with label $S3.b.c$ receives are destined for a node with M_1-index equal to c.

Finally, since these wiring and routing rules hold for all $b, c \in [0, X-1]$, we conclude that: $\forall b, c \in [0, X-1]$, a 3rd-level switch with label $S3.b.c$ receives only packets destined for a node with (M_2,M_1) M-index equal to (b,c). $\square$

Proof of Property 4. Property 2 tells us that a 2nd-level switch with label $S2.a.c$ only receives packets destined for a node with M_1-index equal to c from its leaf-level switches. Thus, we just need to show that the only packets such a switch receives from its 3rd-level switches are also destined for a node with M_1-index equal to c.

According Wiring Rule 2, up-port index b on a 2nd-level switch with label $S2.a.c$ is connected to down-port index a on the 3rd-level switch with label $S3.b.c$ for all $a, b, c \in [0, X-1]$. Thus, for a fixed a, b, and c, the up-port with index b on the 2nd-level switch with label $S2.a.c$ is connected to the 3rd-level switch with label $S3.b.c$. Property 3 then tells us that the 3rd-level switch with label $S3.b.c$ only receives packets destined for nodes with (M_2,M_1) M-index equal to (b,c). Thus, the only packets it may forward down to the 2nd-level switch must be destined for a node with M_1-index equal to c. This is true for all $b \in [0, X-1]$, and so all up-ports on the 2nd-level switch with label $S2.a.c$ will only receive packets destined for a node with M_1-index equal to c.

Finally, since these wiring and routing rules hold for all $a, c \in [0, X-1]$, we conclude that: $\forall a, c \in [0, X-1]$, a 2nd-level switch with label $S2.a.c$ receives only packets destined for a node with M_1-index equal to c. $\square$

4.4 Proof of no contention

We now prove that Node Selection Constraint 2 and Routing Algorithm 2 enable contention-free routing for shift-based communication within a job on a cluster wired according to Wiring Rule 2.

It can be shown that there is no contention between a node and its leaf-level switch using the same arguments presented in Section 3.4. It can also be shown that there is no contention going up from a leaf-level switch to a 2nd-level switch using the same argument presented in Section 3.4 by substituting Y_1 for Y and M_1 for M.

For there to be contention from a 2nd-level switch down to a leaf-level switch, two nodes must be sending packets to nodes on the same leaf-level switch via the 2nd-level switch. Property 4 tells us that all packets received by a given 2nd-level switch are destined for nodes with the same M_1-index. Thus, to have contention, there must be two destination nodes with the same M_1-index on the same leaf-level switch. However, this can not be. The nodes on the same leaf-level switch have consecutive node IDs, which implies they have consecutive M_1-index values. There are no more than Y_1 nodes on a single leaf, and $Y_1 \leq M_1$, so there are not enough nodes on a single leaf-level switch to wrap the M_1-index.

For there to be contention from a 2nd-level switch up to a 3rd-level switch, two nodes from the same 2nd-level leaf must select the same up-port index on the same 2nd-level switch when routing up from a 2nd-level switch to a 3rd-level switch. By Property 4, a given 2nd-level switch receives only packets destined for nodes with the same M_1-index. And by the $S2 \rightarrow S3$ up-port index selection rule, packets select the up-port index that matches the M_2-index of the destination node. Thus, for there to be contention from a 2nd-level switch up to a 3rd-level switch, two nodes on the same 2nd-level leaf must send to destination nodes with the same (M_2, M_1) M-index. The nodes on the same 2nd-level leaf have consecutive node IDs, and in a given step of shift-based communication, all nodes send D hops away. Thus, the corresponding destination nodes in any given step also have consecutive node IDs. There are no more than Y_2 nodes on any 2nd-level leaf, and $Y_2 \leq M_1 * M_2$, so there are not enough nodes on a 2nd-level leaf such that the (M_2, M_1) M-index of the set of corresponding destination nodes can wrap the (M_2, M_1) M-index range.

For there to be contention from a 3rd-level switch down to a 2nd-level switch, two nodes routing packets to the same 3rd-level switch must be sending to destinations on the same 2nd-level leaf. By Property 3, a given 3rd-level switch receives only packets destined for nodes with the same (M_2, M_1) M-index. However, since there are no more than Y_2 nodes on the same 2nd-level leaf, and $Y_2 \leq M_1 * M_2$, there are not enough nodes on any 2nd-level leaf to wrap the (M_2, M_1) M-index range.

As there is no contention between the nodes and leaf-level switches, there is no contention between the leaf-level switches and the 2nd-level switches, and there is no contention between the 2nd-level switches and the 3rd-level switches, Node Selection Constraint 2 and Routing Algorithm 2 implement contention-free routing for shift-based communication within a job on a cluster wired according to Wiring Rule 2.

4.5 Extending to a full three switch level cluster

In a typical production cluster, the goal is to maximize the number of nodes for a given number of switches. For such clusters, the up-ports on the 3rd-level switches would not be left idle. Instead, each of these ports would be assigned as another down-port connected to an additional 2nd-level leaf filled with additional nodes. Although, not proved here, the wiring and routing algorithms can be naturally extended to such clusters, i.e., three switch level clusters containing $2X$ 2nd-level leaves and X^2 3rd-level switches.

In this case, the 3rd-level switches have no up-ports, but instead all $2X$ ports are designated as down-ports with with indices 0 to $2X - 1$. Then, we just extend Wiring Rule 2 to say: $\forall a \in [0, 2X - 1]$ and $\forall b, c \in [0, X - 1]$, connect $S1.a.b : c \leftrightarrow S2.a.c : b$ and connect $S2.a.c : b \leftrightarrow S3.b.c : a$. With this extension, it can be shown using a similar argument as above that Node Selection Constraint 2 and Routing Algorithm 2 implement contention-free routing for shift-based communication within a job on such a cluster.

Taking this further, some clusters may have fewer than X^2 3rd-level switches. Such clusters connect each 2nd-level leaf to the same 3rd-level switch multiple times, e.g., $2X$ 2nd-level leaves and $X^2/2$ 3rd-level switches where each 2nd-level leaf connects twice to each 3rd-level switch. In this case, each 3rd-level switch has multiple down-ports available to reach a given 2nd-level leaf. In order to pick a down-port, we assign each down-port the same (M_2, M_1) M-index that is assigned to the up-port on the 2nd-level leaf that connects to it. Then, when routing packets down, we select the down-port that matches the M-index of the destination node. Although not proved here, with this additional routing condition, the above argument can be extended to show that this implements contention-free routing for shift-based communication within a job on such a cluster.

5 Results

We call this new routing method M-index routing. To verify that M-index routing does indeed provide contention free routing, we wanted to test it on a real system. We created a script that takes the network connectivity information of a system (from `ibnetdiscover`) and the set of nodes a job will run on as input. Then, it outputs a routing table file that can be loaded into OpenSM to program the Infiniband switches. After selecting a set of nodes according to our node selection constraint, we produced the routing tables for Hype, programmed the network using these tables, and then ran mpiGraph. The results of this test are shown in Figure 8 along with the results obtained for the default routing.

With the M-index routing, note how the mpiGraph image has cleared up and is much more uniform. Also, note that all pixels are now centered around a single value in the histogram as opposed to several distinct spikes. This implies that all paths in all steps of the shift-based communication see the same bandwidth. The M-index routing had a dramatic effect on the minimum and average bandwidth values, as well. Whereas with the default routing, the minimum bandwidth was 20% of the maximum and the average bandwidth was 63% of the maximum, with M-index routing, the minimum achieves 82% and the average comes in at 94%. Thus, on average, the network now delivers 94% of its peak instead of just the 63% obtained with the default routing. Although not verified experimentally, we believe this trend should hold for larger systems, as well. In fact, on larger systems, the relative performance improvement will be greater, since, as shown in Table 1, the average bandwidth provided by the default routing decreases with increasing system size. With M-index routing, on the other hand, the average bandwidth should remain essentially constant with increasing system size.

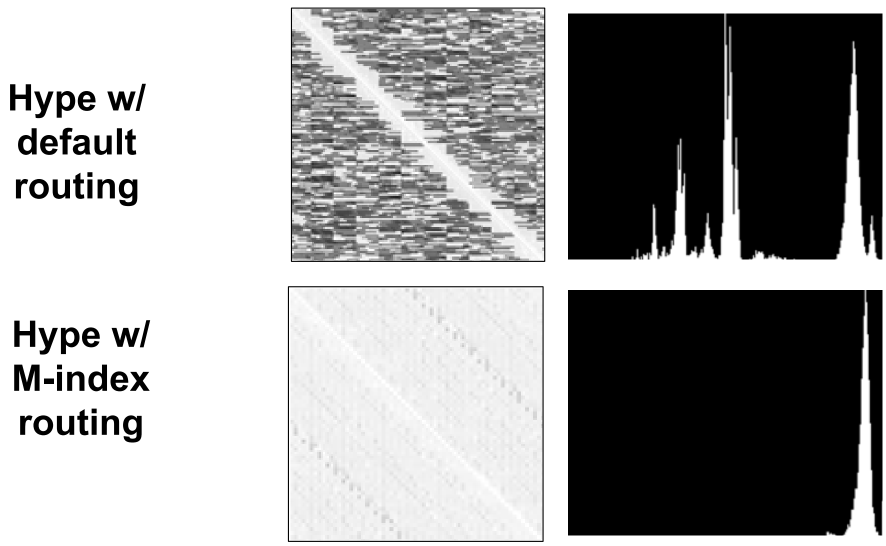

Figure 8: mpiGraph results with default routing and M-index routing on Hype

6 Remaining questions and future work

Others have looked at improving routing in fat-tree Infiniband networks. Some have developed routing algorithms to optimize for random communication patterns [7]. However, in our case, we have a particular communication pattern that is known to be common, and it is possible to develop something that is well tuned to better support this pattern.

Our work generalizes the optimized routing algorithm published in [8]. We enable a more flexible node selection algorithm, which can better account for the down nodes and fragmented node sets that one is likely to encounter on a production machine when scheduling resources for jobs of various sizes and run times while also attempting to maximize resource utilization. However, to achieve this flexibility, one must be able to program the switch routing tables on a per-job basis.

A fair amount of work has been done just to show that a solution exists for contention free routing in shift-based communication within a job, and this paper was written primarily to record this effort. However, this work only leads to a larger set of questions that must be answered to make this approach practical for production clusters.

1. This work assumes MPI processes are assigned to nodes in a certain order with certain limitations. The resource manager that allocates the nodes to a job must be modified to adhere to these conditions. How difficult is it to enforce these conditions for a production resource manager which must also strive for high resource utilization?

2. The routing algorithms discussed in this paper provide contention free routing within a job, but they do not address contention between jobs. How can this work be extended to minimize contention between jobs?

3. This work assumes the network switches can be programmed with routing tables specific to a particular job. Can the switches be programmed with new routing tables without interrupting existing running jobs? How often should the switches be programmed? With each new job, every 10 minutes, every hour?

4. This work assumes that processes will execute the steps of shift-based communication in lock step with one another. What kind of flow control must be added to the MPI collective algorithms in order to prevent processes from advancing to the next step before other processes have completed the current step? And how does one deal with multiple MPI tasks per node?

5. Finally, to what extent does this work help a real application? This work assumes shift-based communication occurs across the global set of processes. However, to be scalable, real applications often divide the global set of processes into distinct subsets and then execute shift-based communication in parallel within those subsets. How can this effort be extended to address this case?

It is the last of those questions we plan to address next. We plan to analyze the communication patterns in some of our key applications, and then we hope to extend this work to optimize routing for communication patterns used within those applications.

References

[1] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," *International Journal of High Performance Computing Applications*, vol. 19, pp. 49–66, 2005.

[2] R. Kumar, A. R. Mamidala, and D. K. Panda, "Scaling alltoall collective on multi-core systems," in *Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8, April 2008.

[3] D. Hensgen, R. Finkel, and U. Manber, "Two algorithms for barrier synchronization," *International Journal of Parallel Programming*, vol. 17, no. 1, 1988.

[4] J. Bruck, C.-T. Ho, C. Kipnis, E. Upfal, and D. Weathersby, "Efficient algorithms for all-to-all communications in multiport message-passing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 11, 1997.

[5] T. I. T. Association, “Infiniband architecture specification: Volume 1, release 1.2,” 2003.

[6] A. Moody, “mpiGraph.” https://sourceforge.net/projects/mpigraph.

[7] X. Yuan, W. Nienaber, Z. Duan, and R. Melhem, “Oblivious routing for fat-tree based system area networks with uncertain traffic demands,” *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 337–348, 2007.

[8] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, “Optimized infiniband fat-tree routing for shift all-to-all communication patterns,” in *Proceedings of the International Supercomputing Conference*, 2007.